# Block successive upper-bound minimization for sparse dictionary learning

Apoorv Goyal
*apoorvg[at]iitk.ac.in*

Satvik Jain
*satvik[at]iitk.ac.in*

***Abstract*—This project aims to explore an algorithmic framework called block successive upper-bound minimization (BSUM). This algorithm is used for Big Data Optimization and is a generalization of popular block coordinate descent (BCD), convex-concave procedure (CCCP), block coordinate proximal gradient (BCPG), nonnegative matrix factorization (NMF) and expectation minimization (EM), etc [1]. In this report we present BSUM and its two other variants - MIBSUM (Maximum improvement upper-bound minimization) and Accelerated-BSUM. We test these methods on the problem of sparse dictionary learning. To the best of our knowledge this is the first application of Accelerated BSUM and MIBSUM for sparse dictionary learning.**

## I. Introduction

### A. Overview of optimization for big data

With advances in modern architecture, modern computers and introduction of new data hungry technologies like Fog computing, computer Vision, image processing, computational biology and social network analysis, large datasets are being generated at a humongous rate. Hence, proper modelling and analysis of such data sets can yield very valuable information for the concerned authorities. However, their shear data size and complexity present a great challenge in their algorithm design and implementation. The general purpose optimization tools that we have been using traditionally are rendered incompetent on the face of theses big datasets. Therefore, Modern large-scale optimization algorithms, especially those that are capable of exploiting problem structures; dealing with distributed, time-varying, and incomplete data sets; and utilizing massively parallel computing and storage infrastructures, have now become the workhorse in the big data era.
To be efficient for big data applications, optimization algorithms must have these certain properties:

- Each of their computational steps must be simple and easy to perform.
- The intermediate results are easily stored.
- They can be implemented in a distributed and/or parallel manner so as to exploit the modern multicore and cluster computing architectures.
- high-quality solution can be found using a small number of iterations.

These requirements preclude the use of computationally expensive operations such as calculating the hessian of a matrix. Therefore, the use of higher order information about the problem is too expensive to obtain.

### B. The BCD method

One of the first variable decomposition methods for solving general minimization problems is the so-called alternating minimization method, which is based on successive global minimization with respect to each component vector in a cyclic order. At every iteration a single block of variables is optimized while the remaining blocks are held fixed. This fundamental method appears in the literature under various names such as the block-nonlinear Gauss–Seidel method or the block coordinate descent method. The Block Coordinate Descent(BCD) method, is a very popular family of optimization algorithms that satisfy most of the aforementioned properties. The basic steps of the BCD are simple and are as follows -

- Partition the entire optimization variables into small blocks.
- Optimize one block variable (or few blocks of variables) at each iteration while holding the remaining variables fixed.

More specifically, consider the following block structured optimization problem -

$$\underset{x}{minimize}\ f(x_1, x_2, \ldots, x_n)\ s.t.\ x_i \in \mathcal{X}_i,\ i = 1, 2, \ldots, n \tag{1}$$

where $f(.)$ is a continuous function (possibly non convex, non smooth), each $\mathcal{X}_i \subset \mathbb{R}^{m_i}$ is a closed convex set, and each $x_i$ is a block variable, $i = 1, 2, \ldots n$. We further define, $x := (x_1, \ldots, x_n) \in \mathbb{R}^m$ such that $\sum_{i=1}^{n} m_i = m$. Let $\mathcal{X} := \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n \subseteq \mathbb{R}^m$. When applying the classical BCD method to solve (1), at every iteration $r$, a single block of variables, say $i = (r\ mod\ n) + 1$, is optimized by solving the following problem -

$$x_i^r \in \underset{x_i \in X_i}{argmin}\ \ f(x_i, x_i^{r-1}) \tag{2}$$

Here, we have defined $x_{-i}^{r-1} := (x_1^{r-1}, \ldots x_{i-1}^{i-1}, x_{i+1}^{r-1})$; for the rest of the variables $j \neq i$, let $x_j^r = x_j^{r-1}$. The BCD algorithm is simple to build and intuitively appealing, but it is incredibly powerful. It is extensively used in a wide range of applications, including signal processing, telecommunication, and machine learning.

These features make the BCD algorithm suitable for large-scale problems where the dimension exceeds the memory and/or the processing capability of the existing hardware.

These features are also illustrated by several applications in signal processing and machine learning, for instance, network anomaly detection and phase retrieval.

## II. Algorithms

### A. The BSUM method

This method is a generalization of the BCD method. An important issue is that it doesn't guarantee convergence when the objective function is non-convex. BSUM circumvents this difficulty by optimizing a sequence of upper-bound approximation to the objective function. Let $u_i(x_i, z) : \mathcal{X}_i \to \mathbb{R}$ be an approximation of $f(x_i, z_{-i})$ for each coordinate block $i$ at a given feasible point $z \in \mathcal{X}$. Then at each iteration $r$, a set $I^r$ is chosen as the set of block variable indices to be optimized, and it is updated by solving the following problem

$$\begin{aligned} x_i^r &\in \arg\min_{x_i \in \mathcal{X}_i} u_i(x_i, x^{r-1}) && \forall i \in I^r \\ x_k^r &= x_k^{r-1} && \forall k \notin I^r \end{aligned} \tag{3}$$

This gives the following Algorithm 1

**Algorithm 1** BSUM Algorithm

1: Find a feasible point $x^o \in \mathcal{X}$ and set $r = 0$
2: **while** Some convergence criterion is not met **do**
3: Pick index set $\mathcal{I}^r$ ▷ $^r$ denotes the $r^{th}$ iteration
4: Let $x_i^r \in argmin_{x_i \in \mathcal{X}_i} u_i(x_i, x^{r-1}), \forall\, i \in \mathcal{I}^r$
5: Set $x_i^k = x_i^{r-1}, \forall\, k \notin \mathcal{I}^r$
6: $r = r + 1$
7: **end while**

At each iteration $r$, we define a set of auxillary variables $\hat{x_i^r}$ as:

$$\hat{x_i^r} \in \underset{x_i \in X_i}{argmin}\ u_i(x_i, x^{r-1}), i = 1, \dots n \tag{4}$$

Following are some of the well known used block coordinate selection rules -

- Gauss-Seidel (G-S) rule : At each iteration $r + 1$, all the index are chosen. Using this rule, all the blocks are updated cyclically with a fixed order.
- Essentially Cyclic (E-C) rule : There exists a given period $T \geq 1$ during which each index is updated at least once i.e.

$$\bigcup_{i=1}^{T} \mathcal{I}^{r+i} = 1, 2, \dots, K, \forall\, r$$

- Gauss-SouthWell (G-SO) rule : At each iteration r, $\mathcal{I}^r$ contains a single index $k^*$ that satisfies

$$k^* \in \left\{ k \;\middle|\; ||\hat{x_k^r} - x_k^r|| \geq q \max_j ||\hat{x_k^r} - x_k^r|| \right\}$$

- Maximum Block Improvement (MBI) rule : At each iteration $r$, $\mathcal{I}^r$ contains a single index $k^*$ that satisfies

$$k^* \in \underset{k}{argmin}\ (x_i^r, \hat{x}_{-i}^{r-1})$$

The approximation function is chosen such that it satisfies the following assumptions

**Assumption A**

A1 $u_i(x_i, x) = f(x),\ \forall x \in \mathcal{X}\ \forall\, i$
A2 $u_i(x_i, z) \geq f(x_i, z_{-i})\ \forall x_i \in \mathcal{X}_i \forall z \in \mathcal{X} \forall i$
A3 $u_i(x_i, z)\ is$ continuous in $(x_i, z)\ \forall\, i$
A4 $u_i^{'}(z_i, z; d_i) = f'(z; d), \forall d = (0, \dots, d_i, \dots, 0)\ s.t\ z_i + d_i \in \mathcal{X}_i, \forall i$

Assumptions A1 and A2 imply that the approximation function is a tight global upper-bound of $f$. Assumption A3 is to ensure that the first order properties of the approximation function and the original function match at the point of approximation. Typically approximation function is chosen such that the subproblems formed have closed form solutions. Some commonly used approximation functions satisfying assumptions A are

Following are some of the commonly used upper bounds satisfying assumption A -

- **Proximal Upper Bound :** Given a constant $\gamma > 0$, adding a quadratic penalization i.e. $\frac{\gamma}{2}||x_i - z_i||^2$ helps set up an upper bound for the function. The proximal operator $prox_f : \mathbb{R}^n \to \mathbb{R}^n$ of $f$ is defined as -

$$\mathbf{prox}_{\lambda f}(v) = \underset{x}{argmin}\ \ (f(x) + \frac{1}{2\lambda}||x - v||^2)$$

Therefore, we define our approximation function as follows -

$$u_i(x_i, z) = f(x_i, z_{-i}) + \frac{\gamma}{2}||x_i - z_i||^2$$

- **Quadratic Upper Bound :** The quadratic upper bound of a function is well known in the literature and is described as follows -
Suppose $f(x) = g(x_1, \dots, x_n) + h(x_1, \dots, x_n)$ where $g$ is lipschitz smooth with $H_i$ as the hessian matrix for the $i^{th}$ block. Then, the bound is given as -

$$\begin{aligned} u_i(x_i, z) = g(z_i, z_{-i}) + h(x_i, z_{-i}) + \\ \langle \nabla_i g(z_i, z_{-i}), x_i - z_i \rangle \\ + \frac{1}{2}(x_i - z_i)^T \mathbf{\Phi}_i (x_i - z_i) \end{aligned} \tag{5}$$

where both $\mathbf{\Phi}_i$ and $\mathbf{\Phi}_i - \boldsymbol{H}_i$ are PSD matrices.
- **Linear Upper Bound :** Suppose $f$ is differentiable and concave, then one can construct

$$u_i(x_i, z) = f(z_i, z_{-i}) + \langle\, \nabla_i f(z_i, z_{-i}), x_i - z_i \,\rangle$$

- **Jensen's Upper Bound :** Suppose $f(x) = f(a_1^T x_1, \dots, a_n^T x_n)$ where $a_i \in \mathbb{R}^{m_i}$ is a coefficient vector and $f$ is convex wrt each $a_i^T x_i$. Let $w_i \in \mathbb{R}_+^{m_i}$

denote a weight vector with $||w_i||= 1$. Then, Jensen's inequality can be rewritten as -

$$u_i(x_i, z) = \sum_{i=0}^{m_i} w_i(j) f\left(\frac{a_i(j)}{w_i(j)}(x_i(j) - z_i(j)) + a_i^T z_i, z_{-i}\right)$$

where $w_i(j)$ represents the $j^{th}$ element in vector $w_i$

Suppose our optimization problem is of the form

$$\min f(x) := g(x_1, \ldots, x_n) + \sum_{i=1}^{n} h_i(x_i),$$
$$s.t \;\; x_i \in \mathcal{X}_i, \; i = 1, \ldots, n \qquad (6)$$

where $g : \mathcal{X} \to \mathbb{R}$ is smooth and for each $i$, $h_i : \mathcal{X}_i \to \mathbb{R}$ is a possible nonsmooth function whose derivative exists at all points in the domain. For such a problem if $\hat{u}_i(x_i, z)$ approximates $g$ then $u_i(x_i, z) = \hat{u}_i(x_i, z) + \sum_{i=1}^{n} h_i(x_i)$ is a valid approximation to $f$. This can be seen as it satisfies all Assumptions A. This form of approximation is used in Sparse Dictionary Learning problem with BSUM.

There are two parts to the main convergence result regarding BSUM. The first one requires the objective function to be quasi-convex, while the second one requires the iterates to lie in a compact set and is thus a stronger result. Below theorem from [2] gives the main result

**Theorem 1.1.** Suppose the cyclic coordinate selection rule is chosen i.e. $\mathcal{I}^r = \{(r \; mod \; n)+1\}$. Let $\{x^r\}_{i=0}^{\infty}$ be a sequence generated by the BSUM algorithm. Suppose assumption A holds, and that each $x^r$ is regular. Then the following holds :

- Suppose that the function $u_i(x_i, y)$ is quasi-convex in $x_i$ for $i = 1, \ldots, n$. Furthermore assume that the subproblem (3) has a unique solution for any point $x^{r-1} \in \mathcal{X}$. Then, every limit point $x^*$ of $\{x^r\}$ is a stationary point of (1).
- Suppose the level set $\mathcal{X}^0 = \{x \mid f(x) \leq f(x^0)\}$ is compact. Furthermore, assume that the subproblem (3) has a unique solution for any point $x^{r-1} \in \mathcal{X}$, $r \geq 1$ for at least $n-1$ blocks. Then, $\{x^r\}$ converges to the set of stationary points i.e.

$$\lim_{r\to\infty} d(x^r, \mathcal{X}^*) = 0$$

where $d(x, \mathcal{X}^*) \triangleq min_{x^* \in \mathcal{X}} ||x - x^*||$ and $\mathcal{X}^*$ is the set of stationary points.

Note that the first part of the theorem requires that there is a unique solution to (3) and it depends on the choice of upper-bound function. In comparison, for convergence the classical BCD method requires that there is a unique solution to (1) which depends only on the objective function. This is an advantage of BSUM over BCD. The second part of the theorem requires regularity of the objective function. This assumption is also necessary for classical BCD method. In general without these assumptions, the convergence is not guaranteed (see [1] for examples and detailed discussion). In section **??**, we see a variant of BSUM where uniqueness of subproblems is not required.

Recently rate of convergence of BSUM type algorithms have been studied extensively. Under the following assumptions the algorithm achieves sublinear rate of convergence [**?**].

**Assumption B**

- B1 The objective is of the form (6) and $g(x)$ has Lipschitz continuous gradient i.e. there exists a constant $L$ such that $||\nabla g(x) - \nabla g(y)|| \leq L||x - y||$, $\forall x, y \in \mathcal{X}$. Both $g$ and $h_i$s are convex functions
- B2 The level set $\left\{x | f(x) \leq f(x^0), x \in \mathcal{X}\right\}$ is compact
- B3 Each $u_i(x_i, z)$ is strongly convex wrt $x_i$

## III. Accelerated BSUM

This method is a variant of the above BSUM method [3]. It is applicable for two block problems of the form (6), i.e.

$$\min f(x_1, x_2) := g(x_1, x_2) + h_1(x_1) + h_2(x_2)$$
$$\text{s.t. } x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2 \qquad (7)$$

The main steps of the algorithm are listed in 2 This method

**Algorithm 2** Accelerated BSUM Algorithm

1: **while** Some convergence criterion is not met **do**
2: Choose $\theta^r = \frac{2}{r+1}$ $\quad \triangleright$ $^r$ denotes the $r^{th}$ iteration
3: $v_1^r = (1 - \theta^{r-1})x_i^{r-1} + \theta^r(w_1^{r-1})$
4: $x_2^r = argmin_{x_2 \in \mathcal{X}_2} f(v_i^r, x_2)$
5: $x_1^r = argmin_{x_1 \in \mathcal{X}_1} u_1(x_1; v_1^r, x_2^r)$
6: $w_1^r = x_1^{r-1} + \frac{1}{\theta^r}(x_1^r - x_1^{r-1})$
7: **end while**

is obtained by applying Nesterov-type acceleration scheme to BSUM with G-S block selection rule. For rate of convergence analysis, we need to make the following assumptions.

**Assumption C**

- C1 Problem (6) is a convex problem, and its global minimum is attained. Also the intersection of domain of $f$ and set of feasible points is non-empty
- C2 Problem $\min_{x_2 \in \mathcal{X}_2} f(x_1, x_2)$ has a unique solution
- C3 The gradient of $g(x_1, x_2)$ with respect to $x_1$ is Lipschitz continuous

Note that here gradient of $g$ wrt to the second block is not required to be Lipschitz continuous. Under these assumptions Acclerated BSUM achieves $\mathcal{O}(1/r^2)$ iteration complexity [3]. It is possible to accelerate BSUM with randomized block selection and quadritic upper-bound having greater than 2 block [4], [5].

## IV. Application to Sparse Dictionary Learning

In many signal processing tasks, it is often required to find a sparse representation of a signal. This is achieved by expressing the signal as a sparse linear combination of signals (called atoms) from a set of signal building blocks (called dictionary). An appropriately chosen dictionary can lead to quite sparse representation of signals. Thus choosing a sparsifying dictionary is an important task in compressive sensing applications.

Formally speaking, given a set of training signals $\{\mathbf{y}_i \in \mathbb{R}^D | i = 1, \ldots, N\}$, our task is to find a dictionary set $\{\mathbf{a}_i \in \mathbb{R}^D | i = 1, \ldots, K\}$ that can sparsely represent the signals in the training set. Let $\mathbf{x}_i \in \mathbb{R}_k, i = 1, \ldots, N$, denote the coefficients of sparse representation of the signal $\mathbf{y}_i$, i.e. $\mathbf{y}_i = \sum_{j=1}^{k} \mathbf{a}_j x_{ij}$, where $x_{ij}$ is the jth component of signal $\mathbf{x}_i$. We define the matrices $\mathbf{Y} \triangleq [\mathbf{y}_1, \ldots, \mathbf{y}_N]$, $\mathbf{A} \triangleq [\mathbf{a}_1, \ldots, \mathbf{a}_k]$ and $\mathbf{X} \triangleq [\mathbf{x}_1, \ldots, \mathbf{x}_N]$. The dictionary learning problem is

$$\begin{aligned} &\min_{\mathbf{A},\mathbf{X}} d(\mathbf{Y}, \mathbf{A}, \mathbf{X}) \\ &s.t. \mathbf{A} \in \mathcal{A}, \mathbf{X} \in \mathcal{X} \end{aligned} \tag{8}$$

where $\mathcal{A}$ and $\mathcal{X}$ are two constraint sets. The function $d(\cdot, \cdot, \cdot)$ measures the goodness of fit of our model. Following is a commonly used function $d(\cdot, \cdot, \cdot)$

$$d(\mathbf{Y}, \mathbf{A}, \mathbf{X}) = \frac{1}{2}||\mathbf{Y} - \mathbf{A}\mathbf{X}||_F^2 + \lambda||\mathbf{X}||_1$$

where $\lambda$ denotes the regularization parameter. The first term is the representation error to keep the estimated signals as close to the training set and second term forces the representation to be sparse. In many applications it is required to constraint the norm of each dictionary atom i.e. $\mathcal{A} = \{\mathbf{A} \mid ||\mathbf{a}_i||_F^2 \leq \beta_i \, \forall i\}$. This leads to the following optimization problem

$$\begin{aligned} \min_{\mathbf{A},\mathbf{X}} \quad & \frac{1}{2}||\mathbf{Y} - \mathbf{A}\mathbf{X}||_F^2 + \lambda||\mathbf{X}||_1 \\ \text{s.t.} \quad & ||\mathbf{a}_i||_F^2 \leq \beta_i \, \forall i \end{aligned} \tag{9}$$

A simple BCD approach to solve this problem by alternatively updating the dictionary $\mathbf{A}$ and the coefficients $\mathbf{X}$ is costly. This problem can be solved efficiently using the BSUM method with quadratic upper-bound. First note that the objective function is of the form (6), thus it is sufficient to find upper-bound approximations to $||\mathbf{Y} - \mathbf{A}\mathbf{X}||_F^2$. This leads to the following subproblems for $\mathbf{A}$ and $\mathbf{X}$

$$\mathbf{A}^{r+1} \leftarrow \arg\min_{\mathbf{A} \in \mathcal{X}} d(\mathbf{Y}, \mathbf{A}, \mathbf{X}^r) \tag{10}$$

Both of the above subproblems have closed form solutions which give the following update rules[1]

$$X \leftarrow S_{\frac{\lambda}{\tau_x}} \left( \mathbf{X} - \frac{1}{\tau_x} \mathbf{A}^T (\mathbf{A}\mathbf{X} - \mathbf{Y}) \right) \tag{11}$$

$$A \leftarrow \mathcal{P}_{\mathcal{A}} \left( \mathbf{A} - \frac{1}{\tau_a} (\mathbf{A}\mathbf{X} - \mathbf{Y}) \mathbf{X}^T \right) \tag{12}$$

where $\tau_x, \tau_a \in \mathbb{R}$ depend on $\mathbf{Y}, \mathbf{A}, \mathbf{X}$ of previous iteration.The notation $\mathcal{P}_{\mathcal{A}}$ is the projection operator to the convex set $\mathcal{A}$ and $S(\cdot)$ denotes the component-wise shrinkage operator i.e. $\mathbf{B} = S_\gamma(\mathbf{C})$ if

$$\mathbf{B}_{ij} = \begin{cases} \mathbf{C}_{ij} - \gamma & \text{if } \mathbf{C}_{ij} > \gamma \\ 0 & \text{if } -\gamma \leq \mathbf{C}_{ij} \leq \gamma \\ \mathbf{C}_{ij} + \gamma & \text{if } \mathbf{C}_{ij} < -\gamma \end{cases}$$

[1]The update rule for $\mathbf{X}$ in Algorithm 2 of [6] has an extra $\mathbf{X}$ term and is incorrect. The update rule we mention here is the corrected one.

As specified in [6], we take $\tau_x = \sigma_{\max}^2(\mathbf{A})$ and $\tau_a = \sigma_{\max}^2(\mathbf{X})$, where $\sigma_{\max}(\cdot)$ denotes the maximum singular value. It is easy to see that this problem satisfies the assumptions in theorem 1.1 and the algorithm thus converges to set of stationary solutions.

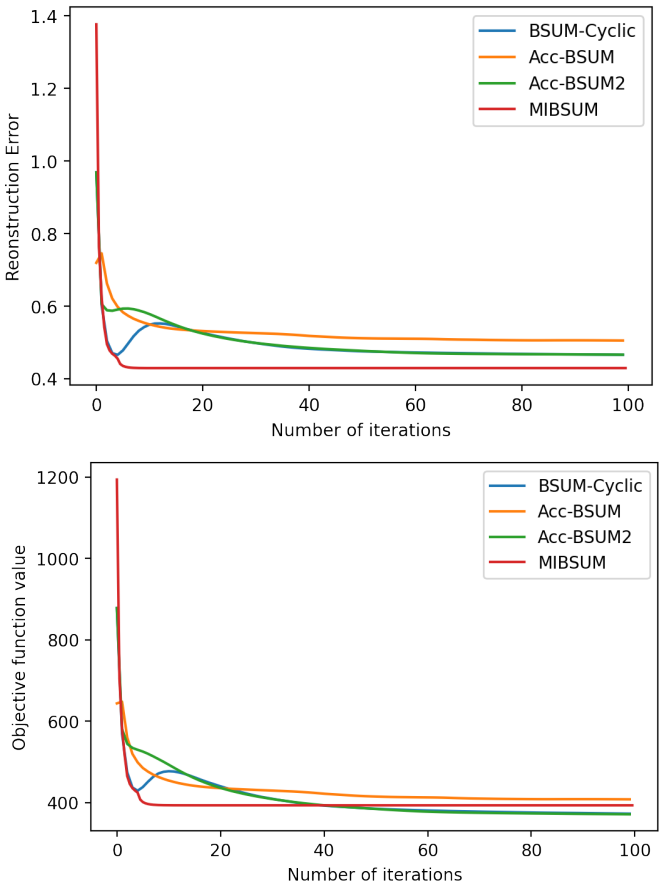


Fig. 1. Methods tested on problem with dimensions $(D, N, K) = (20, 200, 40)$, $\lambda = 1$

Note that the problem is a two-block problem. Take block 1 to be $\mathbf{X}$ and block 2 to be $\mathbf{A}$. Here, Assumptions C are satisfied, thus Accelerated BSUM converges for the problem. We take the same quadratic upper-bound approximation for optimizing $\mathbf{A}$ as done in case of BSUM which also enables us to use closed form solution in 12. The subproblem for optimizing $\mathbf{X}$ is the Lasso problem and for our experiments we used the least angle regression (Lars) to it.

Note that Assumption C2 and C3 are also satisfied if the blocks are interchanged, thus we propose to use upper-bound approximation for block $\mathbf{X}$ as well, to get away with solving the Lasso problem. We use the closed form solution in 11 to update $\mathbf{X}$. We name this Acc-BSUM2.

## V. Simulations

In this section we applied BSUM with cyclic update rule (BSUM-Cyclic), Acc-BSUM, Acc-BSUM2, BSUM with

MBI update rule (MIBSUM) to the problem (9) of sparse dictionary learning. We generated our training set using sklean.make_sparse_coded_signal. Two sets of problem dimensions were taken $(D, N, K) = (15, 100, 20)$. The regularization parameter $\lambda$ was taken to be 1 and 0.5. $\beta_i$s were all taken to be 1. The results are given in Fig 1 and Fig 2.

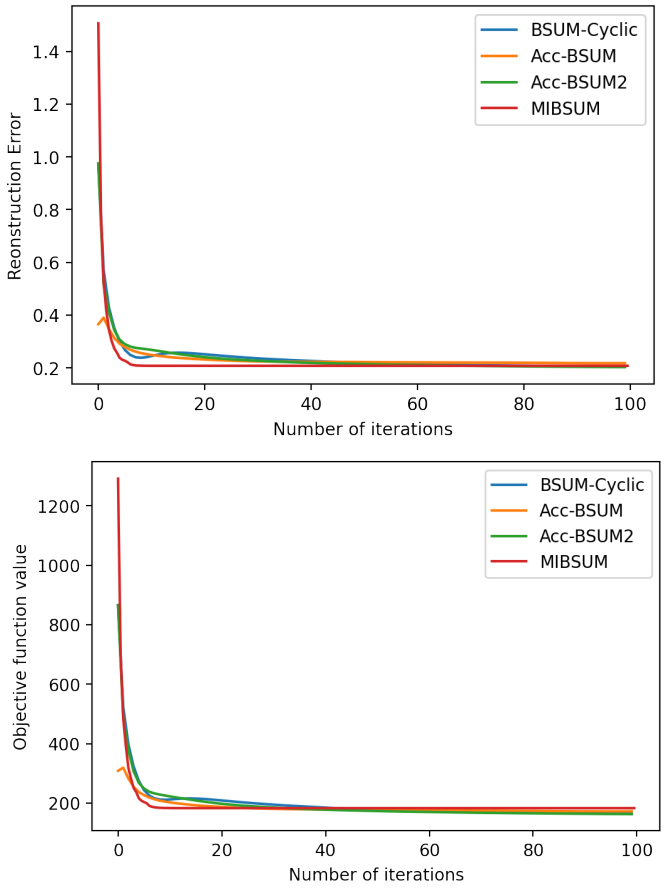


Fig. 2. Methods tested on problem with dimensions $(D, N, K) = (20, 200, 40)$, $\lambda = 0.5$

## VI. Conclusion

It can be seen from the simulations that when the value of regularization parameter is 1, MIBSUM converges in the fewest number of iterations. Acc-BSUM performs better than Acc-BSUM and is also computationally more efficient since we have closed form expressions in the update step. BSUM-cyclic converged faster than Acc-BSUM and Acc-BSUM2. This result was consistent across experiments with different problem dimensions as well. When the value of regularization parameter is 0.5, all algorithms had similar performance. Even though it may appear that MIBSUM converged fastest, but it is not the most efficient due to the fact that all subproblems needs to be solved in each itertion for block selection which is computationally expensive. This is not the case with other methods where first a block is selected then only the subproblem corresponding to that block is solved.

## References


[1] M. Hong, M. Razaviyayn, Z. -Q. Luo and J. -S. Pang, "A Unified Algorithmic Framework for Block-Structured Optimization Involving Big Data: With applications in machine learning and signal processing," in IEEE Signal Processing Magazine, vol. 33, no. 1, pp. 57-77, Jan. 2016

[2] Razaviyayn, M., Hong, M., & Luo, Z. Q. (2013). A unified convergence analysis of block successive minimization methods for nonsmooth optimization. SIAM Journal on Optimization, 23(2), 1126-1153

[3] M. Hong, X. Wang, M. Razaviyayn, and Z.-Q. Luo, "Iteration complexity analysis of block coordinate descent methods," preprint, 2013

[4] Q. Lin, Z. Lu, and L. Xiao, "An accelerated proximal coordinate gradient method and its application to regularized empirical risk minimization," SIAM J. Optim., to be published.

[5] O. Fercoq and P. Richtárik, "Accelerated, parallel and proximal coordinate descent," SIAM J. Optim., vol. 25, no. 4, pp. 1927–2023, 2015.

[6] M. Razaviyayn, H.-W. Tseng, and Z.-Q. Luo, "Dictionary learning for sparse representation: Complexity and algorithms," in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 5247–5251.